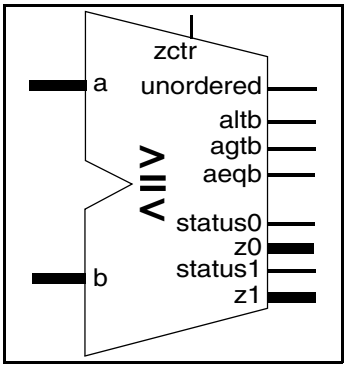
DW_fp_cmp

Floating-Point Comparator

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

- The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- Exponents can range from 3 to 31 bits
- Significand and fractional part of the floating-point number can range from 2 to 256 bits
- Accuracy conforms to IEEE 754 Floating-point standard[1]



Revision History

Description

DW_fp_cmp is a floating-point module that compares two floating-point numbers at inputs `a` and `b`. The format of these numbers is defined by the number of bits in the exponent (*exp_width*) and the number of bits in the significand (*sig_width*). Only one of the four outputs: `agtb`, `altb`, `aeqb` or `unordered` is set as a result of this operation. The input `zctr` is used to select the function that generates the `z0` and `z1` outputs. DW_fp_cmp also produces `status0` and `status1` with information about the `z0` and `z1` outputs. For detailed behavior, see Table 1-5 on page 4 and Table 1-6 on page 5.

Table 1-1 contains pin descriptions and configuration parameters are listed in Table 1-2 on page 2.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*sig_width* + *exp_width* +1 bits	Input	Floating-point number
b	*sig_width* + *exp_width* +1 bits	Input	Floating-point number
altb	1 bit	Output	High when `a` is less than `b`
agtb	1 bit	Output	High when `a` is greater than `b`
aeqb	1 bit	Output	High when `a` is equal to `b`
unordered	1 bit	Output	High when one of the inputs is NaN and *ieee_compliance* = 1
z0	*sig_width* + *exp_width* +1 bits	Output	■ When `zctr` is 0 or open, `z0` = Min(`a`,`b`) ■ When `zctr` is 1, `z0` = Max(`a`,`b`)
z1	*sig_width* + *exp_width* +1 bits	Output	■ When `zctr` is 0 or open, `z1` = Max(`a`,`b`) ■ When `zctr` is 1, `z1` = Min(`a`,`b`)

1. For more information, see IEEE 754 Compatibility in the *Datapath Floating-Point Overview*.

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
status0[a]	8 bits	Output	■ status0[0:6]: Status flags corresponding to `z0`; for details, see STATUS Flags in the *Datapath Floating-Point Overview* ■ status0[7][b]: Set when operand `a` is passed to output `z0`
status1[a]	8 bits	Output	■ status1[0:6]: Status flags corresponding to `z1`; for details, see STATUS Flags in the *Datapath Floating-Point Overview* ■ status1[7][b]: Set when operand `a` is passed to output `z1`
zctr	1 bit	Input	Determines value passed to `z0` and `z1`

a. DW_fp_cmp does not use bit 3 (the Tiny bit) or bit 4 (the Huge bit) of status0 and status1 because rounding is not performed

b. Bit 7 helps to trace the input that has the Max(a,b) or Min(a,b) value, based on the functionality of the `z0` and `z1` outputs (as determined by `zctr`).
For example, when `zctr`=0, `z0`=Min(a,b), `z1`=Max(a,b). So, when a=3 and b=4:
`z0`=Min(3,4)=3 and `status0`[7]=1
`z1`=Max(3,4)=4, and `status1`[7]=0
Looking at `status0`[7] and `status1`[7], you can see that the minimum input value (`z0` output) is coming from input `a` (because `status0`[7] is set), and the maximum input value is coming from input `b`.

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers `a`, `b`, `z0`, and `z1`
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers `a`, `b`, `z0`, and `z1`
ieee_compliance	0 or 1 Default: 0	■ 0: NaNs and denormals are unsupported; NaNs are considered infinities and denormals are considered zeros. ■ 1: The generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_FP_CMP_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_fp_cmp_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_cmp.v	Verilog simulation model source code

The parameter *ieee_compliance* controls the functionality of this component.

- When *ieee_compliance* = 0, the DW_fp_cmp component does not support NaNs and denormals. The `aeqb`, `altb`, and `agtb` outputs are generated considering denormal inputs as zeros, and the `z0` and `z1` outputs pass the values of inputs `a` and `b` without forcing them to zeros. Also, the `unordered` output is always zero in this configuration.
- When the *ieee_compliance* = 1, the component is fully compliant with the IEEE 754 standard and, therefore, operates with NaNs and denormals. The unordered output is set to one when a NaN value is applied as input.

For detailed behavior when *ieee_compliance* = 0, see Table 1-5 on page 4.

For detailed behavior when *ieee_compliance* = 1, see Table 1-6 on page 5.

Table 1-5 DW_fp_cmp Behavior When *sig_width* = 10, *exp_width* = 5, and *ieee_compliance* = 0

a	b	zctr	z0	z1	status0	status1	altb	agtb	aeqb	unordered
0_00011_0000110110 (normal number)	0_00000_0110000000 (denormal: zero)	0	0_00000_0110000000 (min(a,b)= b)	0_00011_0000110110 (max(a,b)= a)	00000001	10000000	0	1	0	0
0_00011_0000110110 (normal number)	0_00000_0110000000 (denormal: zero)	1	0_00011_0000110110 (max(a,b)= a)	0_00000_0110000000 (min(a,b)= b)	10000000	00000001	0	1	0	0
0_00000_0000110110 (denormal: zero)	1_00000_0110000000 (denormal: zero)	0	0_00000_0000110110 (a input)	1_00000_0110000000 (b input)	10000001	00000001	0	0	1	0
0_00000_0000110110 (denormal: zero)	1_00000_0110000000 (denormal: zero)	1	1_00000_0110000000 (b input)	0_00000_0000110110 (a input)	00000001	10000001	0	0	1	0
0_00000_0000000000 (zero)	1_00000_0000000000 (-zero)	0	0_00000_0000000000 (a input)	1_00000_0000000000 (b input)	10000001	00000001	0	0	1	0
0_00000_0000000000 (zero)	1_00000_0000000000 (-zero)	1	1_00000_0000000000 (b input)	0_00000_0000000000 (a input)	00000001	10000001	0	0	1	0
1_11111_0101100111 (NaN: -infinity)	0_10111_0010011010 (normal number)	0	1_11111_0101100111 (min(a,b)= a)	0_10111_0010011010 (max(a,b)= b)	10000010	00000000	1	0	0	0
1_11111_0101100111 (NaN: -infinity)	0_10111_0010011010 (normal number)	1	0_10111_0010011010 (max(a,b)= b)	1_11111_0101100111 (min(a,b)= a)	00000000	10000010	1	0	0	0
0_11111_0101100111 (NaN: +infinity)	1_11111_0010011010 (NaN: -infinity)	0	1_11111_0010011010 (min(a,b) = b)	0_11111_0101100111 (max(a,b) = a)	00000010	10000010	0	1	0	0
0_11111_0101100111 (NaN: +infinity)	1_11111_0010011010 (NaN: -infinity)	1	0_11111_0101100111 (max(a,b) = a)	1_11111_0010011010 (min(a,b) = b)	10000010	00000010	0	1	0	0
0_11111_0101100111 (NaN: +infinity)	0_11111_0010011010 (NaN: +infinity)	0	0_11111_0101100111 (a input)	0_11111_0010011010 (b input)	10000010	00000010	0	0	1	0
0_11111_0101100111 (NaN: +infinity)	0_11111_0010011010 (NaN: +infinity)	1	0_11111_0010011010 (b input)	0_11111_0101100111 (a input)	00000010	10000010	0	0	1	0

Table 1-6 DW_fp_cmp Behavior When *sig_width* = 10, *exp_width* = 5, and *ieee_compliance* = 1

a	b	zctr	z0	z1	status0	status1	altb	agtb	aeqb	unordered
0_00011_0000110110 (normal number)	0_00000_0110000000 (denormal)	0	0_00000_0110000000 (min(a,b)= b)	0_00011_0000110110 (max(a,b)= a)	00000000	10000000	0	1	0	0
0_00011_0000110110 (normal number)	0_00000_0110000000 (denormal)	1	0_00011_0000110110 (max(a,b)= a)	0_00000_0110000000 (min(a,b)= b)	10000000	00000000	0	1	0	0
0_00000_0000110110 (denormal)	1_00000_0110000000 (denormal)	0	1_00000_0110000000 (min(a,b) = b)	0_00000_0000110110 (max(a,b) = a)	00000000	10000000	0	1	0	0
0_00000_0000110110 (denormal)	1_00000_0110000000 (denormal)	1	0_00000_0000110110 (max(a,b) = a)	1_00000_0110000000 (min(a,b) = b)	10000000	00000000	0	1	0	0
0_00000_0000000000 (zero)	1_00000_0000000000 (-zero)	0	0_00000_000000000 (a input)	1_00000_0000000000 (b input)	10000001	00000001	0	0	1	0
0_00000_0000000000 (zero)	1_00000_0000000000 (-zero)	1	1_00000_0000000000 (b input)	0_00000_000000000 (a input)	00000001	10000001	0	0	1	0
1_11111_0101100111 (NaN)	0_10111_0010011010 (normal number)	0	1_11111_0101100111 (a input)	0_10111_0010011010 (b input)	10000100	00000000	0	0	0	1
1_11111_0101100111 (NaN)	0_10111_0010011010 (normal number)	1	1_11111_0101100111 (a input)	0_10111_0010011010 (b input)	10000100	00000000	0	0	0	1
0_11111_0101100111 (NaN)	1_11111_0010011010 (NaN)	0	0_11111_0101100111 (a input)	1_11111_0010011010 (b input)	10000100	00000100	0	0	0	1
0_11111_0101100111 (NaN)	1_11111_0010011010 (NaN)	1	0_11111_0101100111 (a input)	1_11111_0010011010 (b input)	10000100	00000100	0	0	0	1
0_11111_0101100111 (NaN)	0_11111_0010011010 (NaN)	0	0_11111_0101100111 (a input)	0_11111_0010011010 (b input)	10000100	00000100	0	0	0	1
0_11111_0101100111 (NaN)	0_11111_0010011010 (NaN)	1	0_11111_0101100111 (a input)	0_11111_0010011010 (b input)	10000100	00000100	0	0	0	1

Related Topics

- Datapath – Floating-Point Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp.all;
-- If using numeric types from std_logic_arith package,
-- comment the preceding line and uncomment the following line:
-- use DWARE.DW_Foundation_comp_arith.all;

entity DW_fp_cmp_inst is
      generic (
        inst_sig_width : POSITIVE := 23;
        inst_exp_width : POSITIVE := 8;
        inst_ieee_compliance : INTEGER := 0
        );
    port (
    inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_b : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_zctr : in std_logic;
    aeqb_inst : out std_logic;
    altb_inst : out std_logic;
    agtb_inst : out std_logic;
    unordered_inst : out std_logic;
    z0_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    z1_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    status0_inst : out std_logic_vector(7 downto 0);
    status1_inst : out std_logic_vector(7 downto 0)
        );
    end DW_fp_cmp_inst;

architecture inst of DW_fp_cmp_inst is

begin
    -- Instance of DW_fp_cmp
    U1 : DW_fp_cmp
    generic map ( sig_width => inst_sig_width, exp_width => inst_exp_width,
ieee_compliance => inst_ieee_compliance )
    port map ( a => inst_a, b => inst_b, zctr => inst_zctr, aeqb => aeqb_inst,
altb => altb_inst, agtb => agtb_inst, unordered => unordered_inst, z0 =>
z0_inst, z1 => z1_inst, status0 => status0_inst, status1 => status1_inst );

end inst;
```

```
-- pragma translate_off
configuration DW_fp_cmp_inst_cfg_inst of DW_fp_cmp_inst is
 for inst
 end for; -- inst
end DW_fp_cmp_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_cmp_inst( inst_a, inst_b, inst_zctr, aeqb_inst, altb_inst,
         agtb_inst, unordered_inst, z0_inst, z1_inst, status0_inst,
         status1_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 0;


input [sig_width+exp_width : 0] inst_a;
input [sig_width+exp_width : 0] inst_b;
input inst_zctr;
output aeqb_inst;
output altb_inst;
output agtb_inst;
output unordered_inst;
output [sig_width+exp_width : 0] z0_inst;
output [sig_width+exp_width : 0] z1_inst;
output [7 : 0] status0_inst;
output [7 : 0] status1_inst;

    // Instance of DW_fp_cmp
    DW_fp_cmp #(sig_width, exp_width, ieee_compliance)
      U1 ( .a(inst_a), .b(inst_b), .zctr(inst_zctr), .aeqb(aeqb_inst),
          .altb(altb_inst), .agtb(agtb_inst), .unordered(unordered_inst),
          .z0(z0_inst), .z1(z1_inst), .status0(status0_inst),
          .status1(status1_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
October 2021	DWBB_202106.3	■ Added footnote about bits 3 and 4 of `status0` and `status1` in Table 1-1 on page 1 ■ Clarified the releases noted in this table
July 2021	DWBB_202106.1	■ Clarified behavior when *ieee_compliance* = 0 in Table 1-2 on page 2 and in text on page 3 ■ Clarified the purpose of `status0`[7] and `status1`[7] on page 2 ■ Added detailed behavior in Table 1-5 on page 4 and Table 1-6 on page 5
May 2020	DWBB_201912.5	■ Corrected the example in "HDL Usage Through Component Instantiation - VHDL" on page 6
April 2020	DWBB_201912.3	■ Corrected some formatting in this datasheet ■ Added this Revision History table and document links

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com